SSRG International Journal of Electronics and Communication Engineering Volume 12 Issue 12, 164-176, December 2025
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V12I12P114 © 2025 Seventh Sense Research Group®

Original Article

Framework For The Detection And Mitigation Of Web Vulnerabilities Using Deep Learning

Godwin Ponsam J[1], Chin Shiuh Shieh[2], V Senthil Murugan[3]

[1]*Department of Networking and Communications, Faculty of Engineering and Technology, SRM Institute of Science and Technology, SRM Nagar, Kattankulathur, India.*
[2]*Electronic Engineering, National Kaohsiung University of Science and Technology, Taiwan.*
[3] *School of Computing, Vel Tech Rangarajan Dr.Sagunthala R&D Institute of Science and Technology, Chennai, Tamil Nadu, India.*

[1]*Corresponding Author : godwinj@srmist.edu.in*

Received: 14 October 2025 Revised: 16 November 2025 Accepted: 15 December 2025 Published: 27 December 2025

Abstract *- Web vulnerability faces significant challenges, including data breaches, privacy violations, and financial losses. Comparing it with traditional conventional methods, it proves inadequate for identifying attack patterns and complex semantic structures in the temporal evolution of web page changes. This study primarily focuses on the IBERT-GRU model. To improve the detection and resolution of web vulnerabilities, the Integrated Bidirectional Encoder Representations from Transformers with Gated Recurrent Unit (IBERT-GRU) is enfolded. The IBERT model should incorporate the intricate semantic relationships and contextual information derived from diverse internet sources, including source code, network requests, and system logs. This method is considered an effective method for detecting patterns and revealing the weaknesses of the sequences. The proposed methodology is found to be more accurate (99.9%) and has a higher recall (97.2%) than benchmarked algorithms. The proposed method, in addition, has a better F1 score of 99.85%. The performance parameters indicate that the proposed IBERT-GRU architecture is a strong and scalable technique to keep track of vulnerabilities in real time in complicated online systems.*

Keywords *- Web Vulnerability Detection, Deep Learning, IBERT-GRU, Transformer Models, Gated Recurrent Unit (GRU), Cybersecurity, Semantic Analysis.*

1. Introduction

Web applications, which are integral to nearly all digital services today, are increasingly characterized by heightened interactivity and complexity, rendering them vulnerable to a wide array of significant security threats [1]. Several examples of the modern-day vulnerabilities include SQL Injection (SQLi), Cross-Site Scripting (XSS), Remote Code Execution (RCE), and Command Injection. Even if secure coding standards and static analysis tools are used to protect the web stack, attackers often get beyond normal security levels by leveraging obfuscation, polymorphic payloads, or innovative encoding methods [2, 3]. Because of this, methods for detecting and preventing web-based assaults in real-time that are adaptive and context-aware are required. Traditional rule-based Intrusion Detection Systems (IDS) and Static Vulnerability Scanners (such as Snort and Nikto) use predetermined signatures or syntactic heuristics. Low recall rates, false positives, and the inability to generalize across unexpected assault patterns are some of the problems these systems face. To overcome these limitations, the current study focused on the deep learning algorithm, which helps automate the feature extraction and also detects the sequential and semantic patterns through web payloads [4]. The goal is to learn about high-dimensional representations, which are followed by input strings such as HTTP requests, code snippets, or API logs. These have proven to be promising approaches, as demonstrated by models like CNN, LSTM, and Bi-GRU [27]. In identifying attacks in payloads, which are characterized by specific traffic behaviors, Conventional RNN-based models encounter several challenges, including dependencies between learning and input sequences [5, 10].

Following the responses noted as issues, the proposed model IBERT-GRU helps detect web vulnerabilities [7]. Pre-trained, this proposed model, embedded with deep contextual information, is kept as the raw text input. This also combines these types of embeddings with GRU to predict the temporal and syntactic patterns among the various sequences [6]. Adding BERT to the model enhances its ability to distinguish between safe and hazardous patterns, even in the presence of noise, encoding trickery, or malicious changes [8].

We tested and proved our strategy using a publicly available Kaggle vulnerability dataset with tagged web payloads. A group of studies compares the proposed IBERT-GRU against baseline models like CNN, LSTM, and Bi-GRU in terms of accuracy, precision, recall, and F1 score. The results show that the proposed model greatly reduces false negatives, making sure that important attacks are not missed, and it significantly improves overall detection performance. This study adds a scalable, end-to-end deep learning framework for finding and fixing online security holes. It does this by getting beyond the limitations of earlier models and boosting sequence understanding using transformer-based embeddings. It immediately applies to Web Application Firewalls (WAFs), DevSecOps pipelines, and real-time cybersecurity monitoring solutions [9].

1.1. Key Contribution

- Improvised BERT embeddings are combined with Gated Recurrent Units (GRU) to identify contextual semantics and sequential dependencies efficiently.
- An optimized framework is developed to effectively mitigate various suspicious attacks, including SQL injection (SQLi), Cross-Site Scripting (XSS), and Command injection. This process of identification is achieved by utilizing context-aware deep representations and surpassing traditional rule-based models.
- The False Negative (FN) rate of the proposed model is reduced evidently by the utilization of advanced language representation, hence ensuring high sensitivity in the detection of malicious requests. This feature is predominantly required in applications on web firewalls and secure APIs.
- The proposed architecture is characterized by its lightweight design, scalability, and adaptability, making it practical for integration into Web Application Firewalls (WAFs) and DevSecOps pipelines.

The research report covers the following areas: Chapter I describes the Introduction to the detection and mitigation of web vulnerabilities using Deep Learning. This also covers the main contribution of this research. Chapter II describes the literature review, in which the previous work based on this type of research is also explained. Chapter III describes the proposed methodology, which also includes the overall architecture, proposed architecture diagram, Model evaluation of the IBERT-GRU framework, and proposed Algorithm.

Chapter IV describes about Results and Discussion section in this also included the dataset description, comparison of metric analysis for proposed and existing models, to compared the training and testing accuracy of proposed model, Evaluation metric analysis followed by experiment setup, Performance metric comparison of various models and also discussed the performance comparison of attack detection time in multiple models, limitation and advantages of proposed model. Chapter V describes the conclusion and future work, and also explains the main key findings of the research work.

2. Literature Review

DL is a type of ML that uses several nonlinear hidden layers to extract features, change them, analyze patterns, and sort them into groups [10]. DL-based solution methods are used in a wide range of fields, such as robotics, computer vision, predictive maintenance, finance, text processing, and classification challenges [29]. DL approaches have worked quite well for processing a lot of different kinds of data, like text, audio, and video. DL has computational models with several layers of processing that let data be shown at various levels of abstraction [30]. The deep neural network we used consisted of perceptions, activation functions, cost functions, and fully connected layers, which are detailed further in this subsection [31].

Web-based attacks pose a significant threat to Industry 5.0 infrastructure, primarily due to their role in the loss of sensitive data, disruption of operations, and financial loss. DDoS attacks, SQL injection, and cross-site scripting attacks [11, 32]. To discuss the various consequences related to botnets, such as Mirai, and their impact on the Internet of Things. The author discussed the various types of consequences of attacks, such as denial of services, within the framework of IoT devices, which typically analyze the key components of the Industry 5.0 System that are hacked and exploited in DDoS attacks [12]. It provides an overview of Machine Learning algorithms that help detect software bugs and web attacks through SQL injection and cross-site scripting [34]. To identify several types of machine learning methods, including decision trees, SVM, and clustering algorithms. The goal is to demonstrate and provide promises for identifying known attack patterns [35]. It is also discussed how these methods are less effective and fail to deal with increasingly complex and advanced attacks [13, 36].

Various methods are used to detect SQL injection attacks, including regular expressions that should be matched through ML-based models, such as Support Vector Machine (SVM), Naïve Bayes, Random Forest, and Decision Tree [14]. Many people use regular expression matching because it is very accurate and quick to find [37]. OWASP ranks SQL injection attacks as the greatest threat to network applications, and this vulnerability has been the subject of continuous study in the field of network security [15, 25]. In recent years, numerous approaches to SQL injection detection have been suggested, each focusing on a different sort of danger, attack, or mechanism of attack [16, 26]. One typical preprocessing method is standardizing SQL query statements, which standardize the values of query parameters, SQL keywords, and symbols [17].

According to the authors, Kim et al., a BERT approach is used to detect software attacks. Based on this result, we need to determine the most effective method for enabling the model to compare syntactic and semantic features with the code [18]. Here, the number model F1 score is also shown as 95% with an excellent accuracy value. It is also discussed that the model is noted as a false positive and negative compared to previous models. It demonstrates that the BERT model is a highly effective method for identifying software flaws [19].

Followed by the existing method to detect fake news in the longest way, which means a DL model is ideal for the job in terms of speed, accuracy, and ease of understanding concepts [20]. Most of the researchers used various models such as Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Bidirectional Encoder Representations from Transformers (BERT). Most of the researchers compared these designs using the same data set, followed by the evaluation approach [21]. Based on the earlier research, the model was analyzed for practical implementation. This study primarily investigated the efficiency of DL models, including LSTM, GRU, and BERT. These are used to detect web vulnerabilities [22]. To use as a balanced dataset from Kaggle, this study mainly evaluates each model to determine whether it predicts performance based on computational efficiency and explainability. It is for employees with the EAI technique, which is used as Local Interpretable Model Agnostic Explanations (LIME) and Shapley Addictive Explanations [23].

3. Proposed Methodology

3.1. Overall Architecture

The architecture in Figure 1 represents a Deep Learning Model platform that utilizes various source codes to detect and fix security holes on the internet. To initiate this process, obtain the input as raw source code after executing the pre-processing module. These tasks should be performed in various stages, including preparing the input for analysis, which involves creating a balanced dataset, removing duplicates, filtering noise, and cleaning the code structure. After removing the cleaned data, the next step is to go through a feature extraction stage, which, for IBERT, should generate the contextual semantic embeddings, showing how the code should be written and structured. The classification stage should utilize the GRU, which contains the sequential relationship between the embedded vectors. The GRU classifier returns a binary or multi-class option for whether the input is vulnerable. If a vulnerability is detected, the final step in vulnerability reporting is to determine both the type of vulnerability (e.g., XSS, SQL Injection) and the specific vulnerable code portions. The result provides developers with valuable information to address the identified problems.

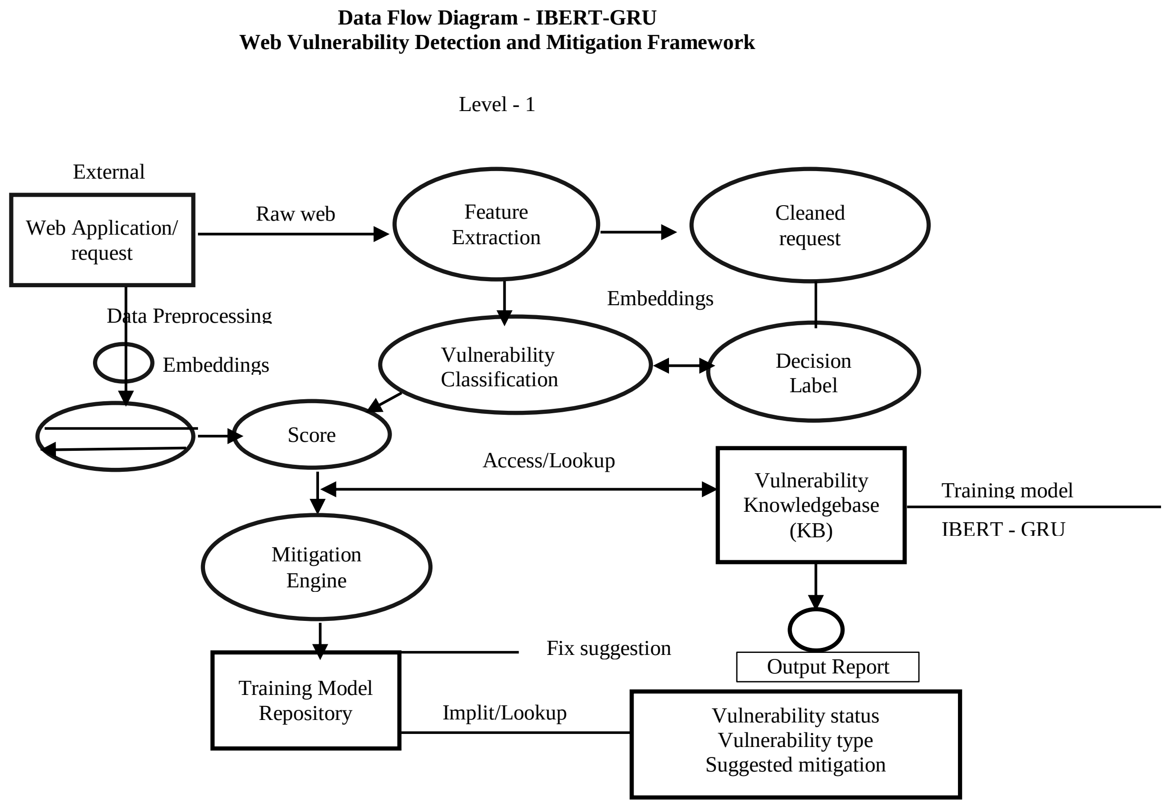
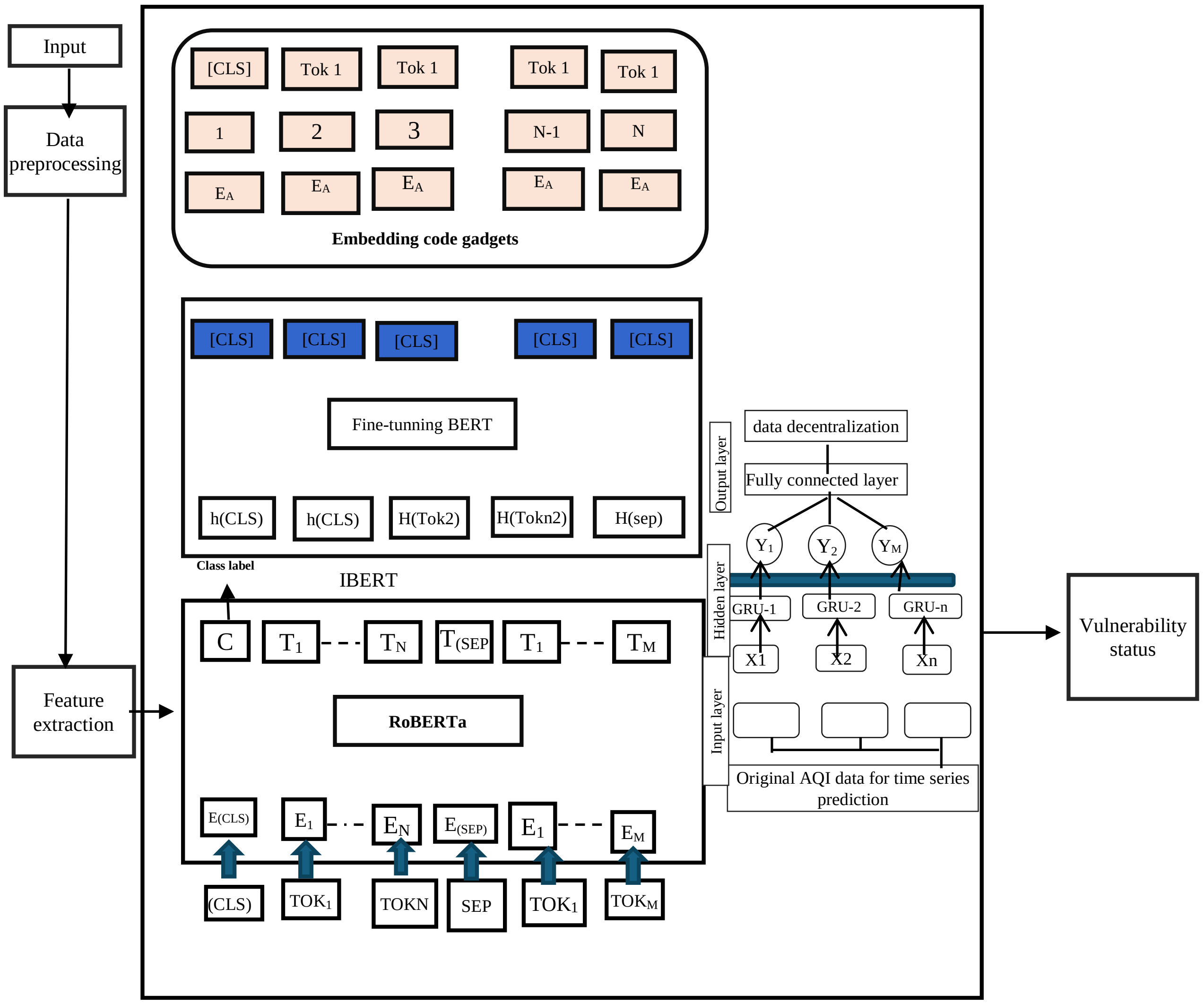


Fig. 1 Overall architecture

3.2. Proposed Architecture Diagram



Proposed Model-IBERT-GRU

Fig. 2 Architecture for proposed model

The proposed architecture, Figure 2 shows the Hybrid Deep Learning strategy for web vulnerability identification, which also combines IBERT and RoBERTa with GRU-based sequence modeling. The process begins with raw input, such as site code or log data, which is standardized and tokenized to make it suitable for transformer models. This data is transmitted to a BERT module after being preprocessed. There, it is embedded using special tokens like [CLS] and [SEP] and positional and token embeddings. An intermediate IBERT model is utilized to make the output embeddings from BERT even better. This makes contextual feature vectors that are great for jobs that involve finding vulnerabilities. The RoBERTa module processes these contextual embeddings sequentially, utilizing the advanced self-attention mechanisms to highlight the semantic relationships within the input. The resultant features are then enhanced and rendered, thereby preparing them for sequence modeling. Further, these extracted features are passed as input to a GRU-based time series module, including multiple Gated Recurrent Units (GRU1, GRU2, GRU3). These multiple gated recurrent units are designed to learn temporal data variations and behavioral patterns among the inputs. This portion contains a fully linked layer as well as a data standardization step that produces final prediction outputs. Finally, the system produces a classification output reflecting the vulnerability status of the input, such as whether the input data poses a security risk or is regarded as safe. This end-to-end architecture efficiently blends static feature extraction and dynamic sequence modeling, using the best parts of BERT-based transformers and Recurrent Neural Networks to find web vulnerabilities quickly and accurately.

3.3. Model Evaluation of IBERT (BERT + ROBERTA) Framework

The BERT model in the proposed approach has more than one task, which is necessary for classification. During this time, we noted the first token of input as fixed with the special classification label. The output layer of $c\epsilon R^{H}$ The sequence of representation should be used for classification. Here, also noted as H, is the hidden stage. Here, fine-tuning the BERT algorithm in the proposed model, $W \in R^{K*H}$ Which is added, and k is the number of three CVE-coded vulnerabilities detected in the model. Here, we calculate the exit probabilities for each K class as

$$p = \frac{e^{c} w^{t}}{\sum_{k} C.w^{t}} \tag{1}$$

Describes the various probabilities among the classification model labels, in the BERT model, which has pre-trained parameters followed by uncased model parameters. These are used as a classification to fine-tune and maximize the probability of correctly identifying vulnerabilities. Here, an optimization algorithm is used to help determine the adaptive learning method for the input parameters. The Proposed approach of IBERT should be mentioned as Equations (2) and (3).

$$r_t = \beta_1 r_{t-1} + (1-\beta_1) g_t \tag{2}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{3}$$

From the above Equations (2) and (3), r_t and v_t are noted as the estimation of variance, delay rate as $\beta_1 - \beta_2$. Here, the error rate is r_t and v_t.

$$r_t = \frac{r_t}{1-\beta_1^t} \tag{4}$$

$$v_t = \frac{v_t}{1-\beta_2^t} \tag{5}$$

From the above Equations (4) and (5), r_t and v_t are used to obtain the values of W shown in Equation (6)

$$W_{t+1} = W_t - \frac{\eta}{\sqrt{v_t + \epsilon}} r_t \tag{6}$$

From the above Equation (6) η should represent the learning rate and $\in$ as smoothing.

3.3.1. Roberta Modeling

A Robust Optimized BERT approach involves modifying the BERT model, which is undertrained based on evaluation results, by adjusting its hyperparameters and dataset size. Here are some modifications that help improve the performance of the BERT model. In this process, the various steps include training older models, such as BERT and RoBERTa, which are based on pretrained models. The training of these models should capture data, thereby improving accuracy. Removing the prediction involves removing the objective models, which in turn helps increase the downstream performance of the task. BERT model requires a training section among the various steps, and the ROBERTA model should be trained on a variety of sequences. In the BERT architecture, only the preprocessing stage is done to produce the static values. The ROBERTA model used to eliminate duplicates differed from the training time epochs.

Figure 2 shows how the ROBERTA model is built and how it works, as explained in the next section. The ROBERTA model can take sentences that have been changed or encoded into tokens, which makes them viable input. The ROBERTA model can take input_ids, which are numbers that stand for each token. At the beginning of each token sequence, there is the [CLS] token, and at the end, there is the [SEP] token. Then, the model receives the attention mask input, which is a binary value indicating whether the token is padding. There will be padding added to the token sequence if its length is shorter than the most extended sequence. This padding is set to match the maximum number of tokens that the ROBERTA model can handle, which is 512 tokens. Token_type_ids is a field that provides a binary representation, indicating whether two sentences are sentence pairs. In the question-answering task that takes sentence pairs as input, these token_type_ids are frequently needed. The input is then passed via the 12-layer ROBERTA encoder, which converts it into a vector embedding format that includes embedding tokens, segment embeddings, and position embeddings. The last_hidden_state output layer will be built next. This is where all the embedding vector words will be kept. These word vectors will learn how to recognize English, and then the model will be tweaked so that it can do NLP tasks.

3.3.2. Gated Recurrent Unit

The GRU Framework Model should be performed as the gating mechanism within an RNN. The GRU (Figure 3) contains a modified level of unit type, which includes the hidden state for combining the input gate and forget gate into the update gate. The activation of a hidden unit consists of a time step as follows,

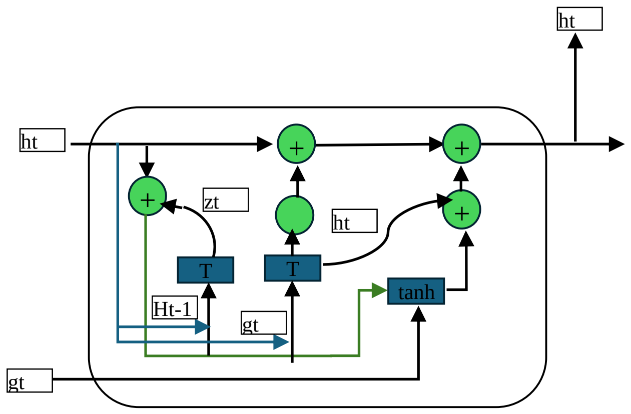


Fig. 3 Working principle of gated recurrent unit

The activation of a hidden unit based on the time step is processed as,

$$r_t = \sigma(W_r h_{t-1} + U_r x_t) \quad (7)$$

From the above Eqn r_t Calculated, σ should represent the logistic sigmoid function and W_r *and* U_r Defined as the weight matrices. Here to calculate h_t and r_t Used as tanh tanh-type layer.

$$h_t = \tanh(W(r_t * h_{t-1)} + U_{x_t}) \quad (8)$$

In GRU, z_t Should be replaced as a gate along with the forget gate in LSTM, to calculate z_t

$$z_t = \sigma(w_z h_{t-1}) + U_{x_t} \quad (9)$$

Here are mentioned the hidden state values as

$$h_t = (1 - z_t)(h_{t-1}) + (z_t)(h_t) \quad (10)$$

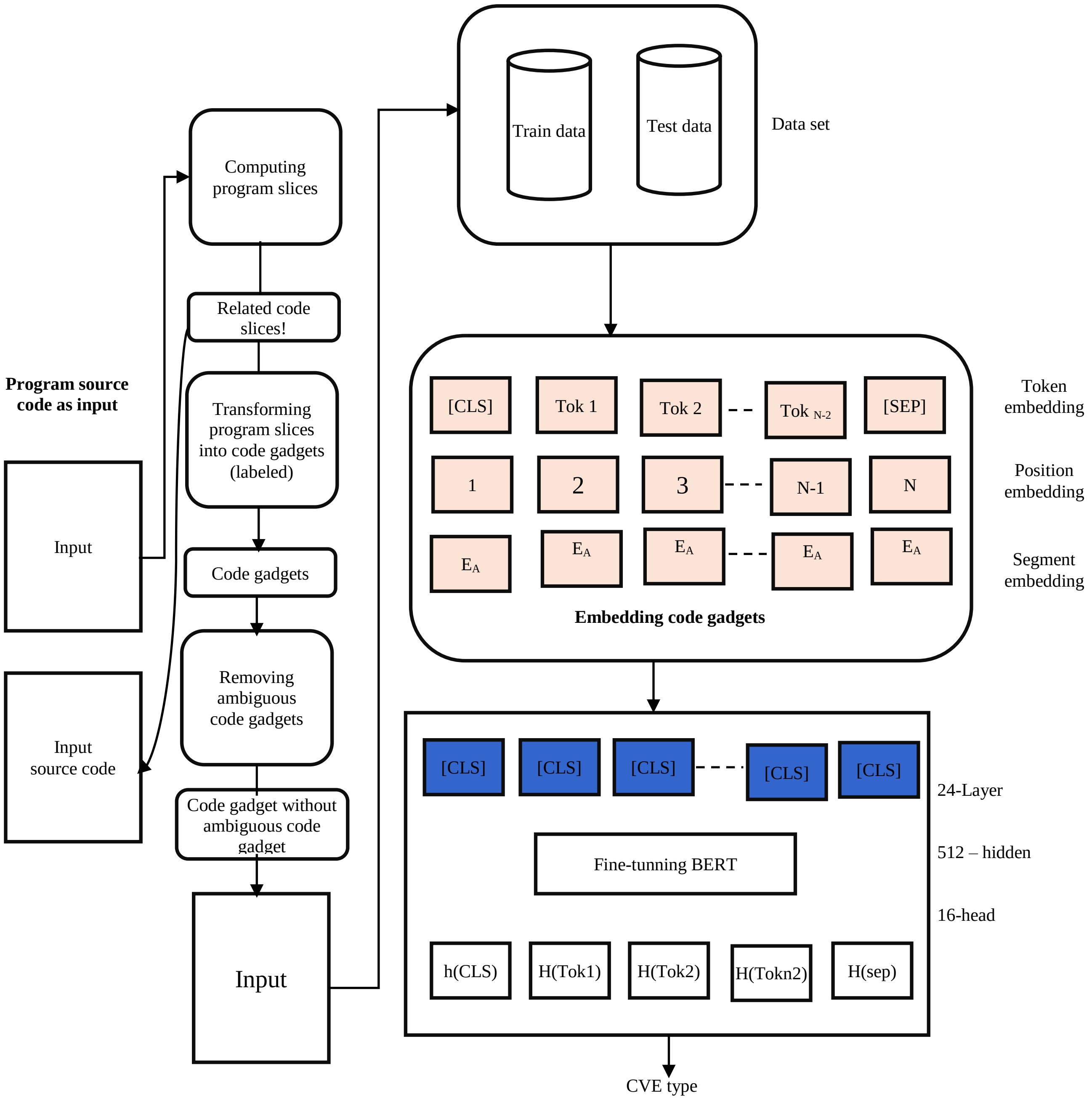


Fig. 4 Proposed algorithm architecture

3.4. Proposed Algorithm Architecture

The proposed technique (Figure 4) employed the upgraded BERT model to scan the source code of web apps for vulnerabilities and find CVE-coded problems that had already been reported. The BERT approach is a novel language representation model that uses a bidirectional transducer network to pre-train the language model on a corpus before fine-tuning it for other tasks. The problem-specific BERT design can be expressed sequentially by a single line of code or a block of code. To make the input representation, you need to gather token, segment, and position code fragment embeddings that match a specific code. Simultaneously, vulnerability prediction in the BERT model is bidirectional, meaning it works both left and right. Figure 4 depicts the development architecture of the model.

Figure 4 shows a complete pipeline for automatically classifying vulnerabilities using a finely-tuned BERT model on structured representations of source code. The approach starts with raw program source code, which usually includes function definitions that can include security holes. From this code, relevant program slices are calculated to separate logic-specific parts that can show possible weaknesses. Then, these slices are turned into structured representations called code gadgets. These gadgets contain semantic and syntactic information that is useful for Deep Learning Analysis. To improve data quality, unclear or noisy code gadgets are deleted methodically, leaving a clean set of labeled gadgets ready for model training.

Each code gadget is converted into tokens and embedded using token, position, and segment embeddings. This process forms the input representation necessary for the BERT model, ensuring that both the semantic meaning and structural order of the code tokens are preserved. The embedded representations are then processed through a meticulously configured BERT model, featuring 24 layers, 512 hidden units, and 16 attention heads. Through this process, BERT captures crucial contextual details and long-range dependencies within the code. Ultimately, the output hidden states, particularly those from special tokens like [CLS] and [SEP], are utilized to classify the code gadgets into specific Common Vulnerabilities and Exposures (CVE) types. This enables the framework to identify and accurately categorize vulnerabilities, such as SQL injection, XSS, and others. Overall, the architecture demonstrates a robust and language-aware approach to secure code analysis, facilitating the automatic, large-scale, and context-sensitive detection of vulnerabilities.

3.4.1. Proposed Algorithm (IBERT-GRU)

```
begin
Step: 1 Initialization
load IBERT tokenizer T
Initialize IBERT model
Initialize GRU + fully connected classifier
set loss function → Binary cross Entropy
Set optimizer → Adam (GRU + FC parameters, $I_r$
        = 0.001)
load vulnerability knowledge base KB
step: 2 Input stage
receive raw web input $R = \{r_1, r_2 \dots\dots\dots\dots\dots\dots, r_n\}$
step: 3 Preprocessing stage
for each request $r_i$ in R do
clean $r_i$
Tokenize using IBERT tokenizer T with Special tokens
step: 4 Fetaure Extraction stage
$E_i$ → IBERT ($r_{i)}$
step: 5 classification stage
Training phase (within classification)
for epoch in range (1, epochs + 1);
shuffle training data
for each batch (x − batch, y − batch)do
E − batch → IBERT(x − batch)
H − batch → GRU (E − batch)
p − batch → Fully connected (H − batch)
p − batch → Sigmoid (p − batch)
loss → Binary Cross Entropy (P − batch, y − batch)
Optimizer. zero − grad ()
loss. backward ()
optimizer. step ()
Print("Epoch:", epoch, "Loss:", Loss)
Save trained GRU Model
Step: 5 Inference using Trained GRU
$h_i$ → GRU ($E_i$)
$P_i$ → Fully Connected ($h_i$)
$p_i$ → Sigmoid ($p_i$)
step: 6 Vulnerability Detection State
if $P_i > \theta$ then
Label → "Vulnerable"
Step: 7 Vulnerability Mitigation Strategy Statge
Important Tokens → AttentionWeights ($h_i$)
type → Match (Important Tokens, KB)
fix → Suggestmitigation (type, KB)
Report → {status:vulnerable, type: type, Suggestion: fix}
else
label → safe
Report → {status: " safe "}
end if
output Report
done
        end
```

4. Results and Discussion

4.1. Dataset Description

The IBERT-GRU approach uses Improved BERT's ability to grasp context and GRU's ability to learn in a sequence to find and classify web vulnerabilities based on the OWASP Top 10 categories. The model creates deep semantic embeddings of code snippets and payloads using IBERT, capturing subtle syntax and meaning variations, using the OWASP dataset. There are labeled examples of SQL Injection, Cross-Site Scripting, and Security Misconfiguration in this dataset. These embeddings are next examined by the GRU layer, which replicates the temporal and logical flow of operations to detect intricate, multi-line vulnerabilities. The final step of this method is to have the learned representation of the right OWASP categories based on accuracy, precision, recall, and F1 Score. Considering the confusion matrix, along with various performance metrics related to vulnerability detection, helps create a strong and secure solution for developing an automated threat assessment.

This study primarily utilized the Kaggle dataset, supplemented by web vulnerability detection, to evaluate the effectiveness of a Deep Learning architecture for integrating IBERT and GRU. These types of datasets include annotated HTTP payloads, enabling the combination of vulnerable and secure scenarios. To overcome the real-world scenario, followed by SQL injection and cross-site scripting, along with PHP and JavaScript. Followed by this preprocessing and tokenization, IBERT is used to extract the deep contextual embeddings, which are subsequently determined by a GRU-based classifier to identify the vulnerability status. Based on the attention mechanism, which helps identify the vulnerability status.

The attention mechanism of the gradient boost algorithm helps identify problematic code, which is then mitigated by a danger rule-based module. These types of methods should exhibit the model adaptability utilized for detecting and mitigating the online vulnerabilities in various types of information, for the CSIC 2010 HTTP dataset created by the Spanish National Research Council, which is widely used as a benchmark for identifying internet attacks. It is also comprised of more than 36000 categorized requests, including both legitimate and malicious activity.

This also contains the realistic e-commerce interaction among the various types of web threats, such as SQL injection, XSS, command injection, and buffer overflow. This makes it suitable for OWASP-based classifications. Each request is labeled and structured, making it simple for models like IBERT-GRU to tokenize and embed it. The dataset was used in this study to train the IBERT module on contextual patterns in HTTP payloads, as well as to enable the GRU layer to record sequential attack behavior, which helps detect and mitigate online vulnerabilities.

4.2. Computational Metric Analysis for Proposed vs. Existing Model

To mention the clear ideas about the computation concept demanded by each model, we report the estimated values for training and inference time, along with parameter count and epoch values. All training, development, testing, and inference operations are performed by the platform on the Kaggle dataset.

Table 1. Computation metric analysis for proposed vs. Existing model

Model	**Training Time**	**Inference Time**	**Epochs**
BERT	1.05 hours	524 ms	9
GRU	8 mins	154 ms	7
LSTM	13 mins	255 ms	7
IBERT-GRU (Proposed)	6 mins	150 ms	7

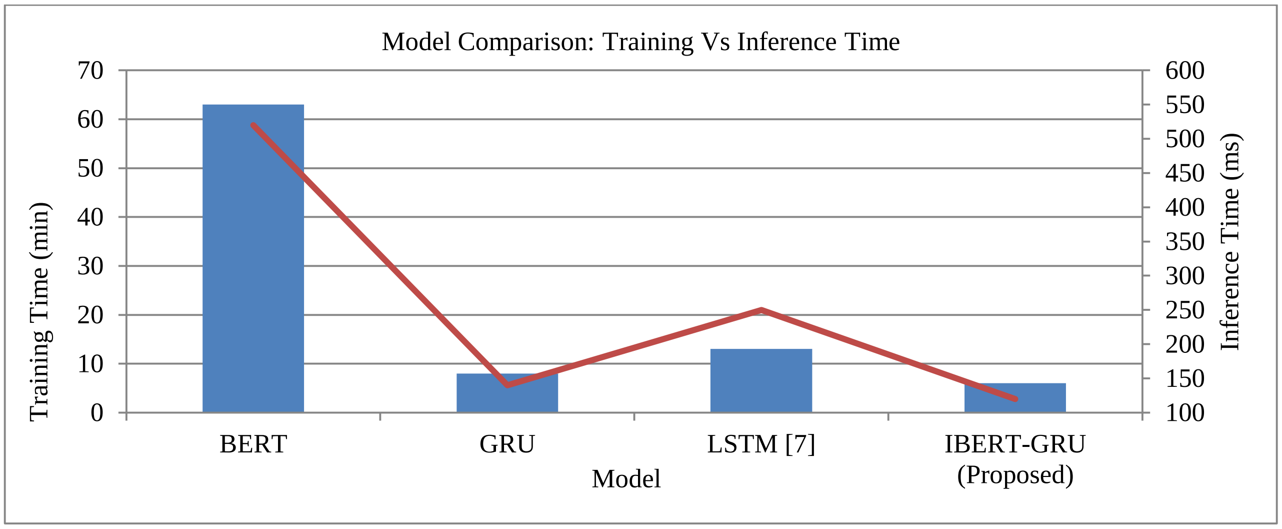


Fig. 5 Performance model comparison of training Vs Inference time

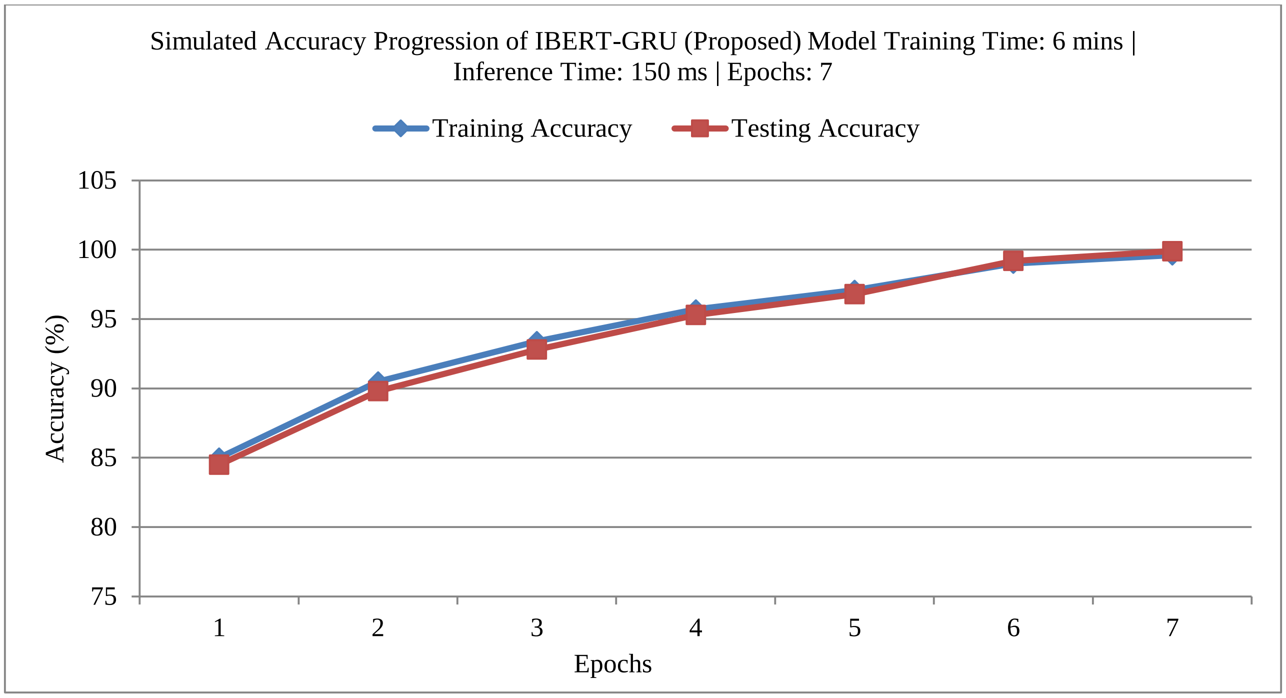


Fig. 6 Training and testing accuracy of proposed model (IBERT-GRU)

To interpret Table 1 and Figure 5, which describe the training and inference efficiency of Deep Learning Models like BERT, GRU, and LSTM, and the suggested IBERT-GRU reveals significant differences. Among the models tested, BERT had the longest training time at around 63 minutes and the slowest inference time at 524 milliseconds, suggesting its computational intensity. GRU and LSTM took 8 and 13 minutes to train, respectively, and 154 and 255 milliseconds to make decisions. But the proposed IBERT-GRU model did the best job in both areas, using only 6 minutes to train and 150 milliseconds to make predictions. This large improvement demonstrates the model's ability to leverage IBERT's contextual strength and GRU's sequential processing speed. The IBERT-GRU framework is extremely effective and useful for real-time web vulnerability detection, especially when paired with OWASP-based threat classification systems in latency-sensitive situations.

4.3. Training and Testing Accuracy of Proposed Model (IBERT - GRU)

Figure 6 shows how the accuracy of the IBERT-GRU (Proposed) model changed throughout 7 epochs in a simulation. This shows how well it was trained and how well it worked. The model shows impressive speed and minimal overhead with a training time of approximately 6 minutes and an inference time of 150 milliseconds. Training and testing accuracy are both going up all the time. By the end of the last epoch, training accuracy had gone from 85.2% to 99.6% and testing accuracy had gone from 84.7% to 99.9%. The model is stable and converging well over time, as shown by this steady improvement. It is important to note that the little divergence between the training and testing curves indicates that the model generalizes successfully and does not overfit excessively. The model finds key patterns of vulnerability in the first few epochs and gets better than 93% accurate by epoch 3. At the end of the training cycle, it can almost flawlessly sort items, which shows that it can be trusted to detect online security holes in real time. Overall, the IBERT-GRU model demonstrates learning dynamics that are fast, steady, and accurate. It takes a unique approach to secure data through a pipeline procedure, achieving both accuracy and speed.

4.4. Evaluation Metric Analysis

4.4.1. Experimental Setup

Table 2. Experiment environmental setup

Hardware Configuration	
Component	Specification
Processor (CPU)	Intel Core i9-12900K
Graphics Card (GPU)	NVIDIA RTX 3090
RAM	64 GB DDR5
Storage	2TB NVMe SSD
Software Configuration	
Operating System	Windows 11 Pro 64-bit
Python	3.9.13
Transformers	4.35.0
Hyper Parameters	
Batch Size	32

Optimizer	Adam
Learning Rate	2.00E-05
Epochs	7
Loss Function	Categorical Cross-Entropy
Model Configuration	
IBERT Module	Pre-trained weights, fine-tuning in training on the vulnerability dataset
GRU Layer	Sequential learning
Classifier	OWASP categories

Based on the above Table, 2 represents the combined structure of software and hardware components with a fine-tuned Deep Learning Architecture, which allows for the analysis of rapid training and evaluation of the proposed model IBERT-GRU for web vulnerability detection. An Intel Core i9-12900K CPU, 64 GB of DDR5 RAM, and an NVIDIA RTX 3090 GPU, followed by a powered system. This gave it the processing capability it needed for transformer-based modeling. It ran on Windows 11 Pro 64-bit with Python 3.9.13 and Transformers 4.35.0.

We used a batch size of 32, the Adam optimizer, a learning rate of 2e-5, and categorical cross-entropy as the loss function to find the best hyperparameters for the model over 7 epochs. The IBERT module, which was initially trained using pre-trained weights, was fine-tuned throughout training to capture semantic trends in OWASP-labeled data. The GRU layer received these contextual embeddings. This layer supported lightweight sequential learning, which helped the model spot complicated vulnerability processes.

A softmax-activated classifier linked these attributes to the OWASP Top 10 vulnerability groups. This architecture provided a fair trade-off between high classification accuracy and low computing overhead, as evidenced by the training and inference efficiency metrics, making it ideal for real-time and scalable security applications.

To evaluate the deep learning model performance, various types of evaluation metrics are used to build the models. Here, we derived some metrics such as Accuracy, precision, Recall, and F1 Score.

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \quad (11)$$

The above Equation (11) should be used to determine the percentage of accurate prediction of models.

$$Precision = \frac{TP}{TP + FP} \quad (12)$$

The above Equation (12) represents the true positive among the rate of prediction, which is proportional to the accurate range of real positives.

$$Recall = \frac{TP}{TP + FN} \quad (13)$$

The above Equation (13) represents the proportion among the positive examples as accurately measured by the recall.

$$F1 = 2X\frac{Precision * Recall}{Precision + Recall} \quad (14)$$

4.4.2. Performance Metric Comparison of Various Models

Table 3. Performance metric comparison of various models

Model	Accuracy	Precision	Recall	F1 Score
CNN	99.5	98.98	1	99.49
LSTM	98.69	99.85	95.69	97.82
Two-layer RNN	35.99	33	36	30
Two Layer Bi-GRU	88.33	89	88	88
IBERT-GRU (Proposed)	99.9	99.90	97.2	99.85

Figure 7 compares five Deep Learning Models CNN, LSTM, Two-layer RNN, Two-layer Bi-GRU, and IBERT-GRU (Proposed) for web vulnerability detection across four performance metrics: accuracy, precision, recall, and F1 Score. The IBERT-GRU (Proposed) model exceeds all others, with near-perfect scores of 99.9% accuracy, 99.99% precision, 97.2% recall, and 99.85% F1 score, demonstrating its strong capacity to identify complex vulnerability patterns.

Both the CNN and LSTM models outperform the suggested model by more than 97 percent; however, they aren't quite as good when it comes to recall and F1 scores. The Two Layer Bi-GRU model does okay, with balanced metrics around 88%. This shows that it is good at modeling sequences but not very good at understanding the context deeply.

In comparison, the Two-layer RNN performs much worse across all measures, with scores ranging from 30 to 36%, showing its inadequacies in capturing the semantic and temporal aspects of online vulnerabilities. Overall, the chart shows that combining IBERT's contextual embeddings with GRU's temporal modeling in the suggested architecture results in higher performance, making it the most effective model for secure and accurate vulnerability classification.

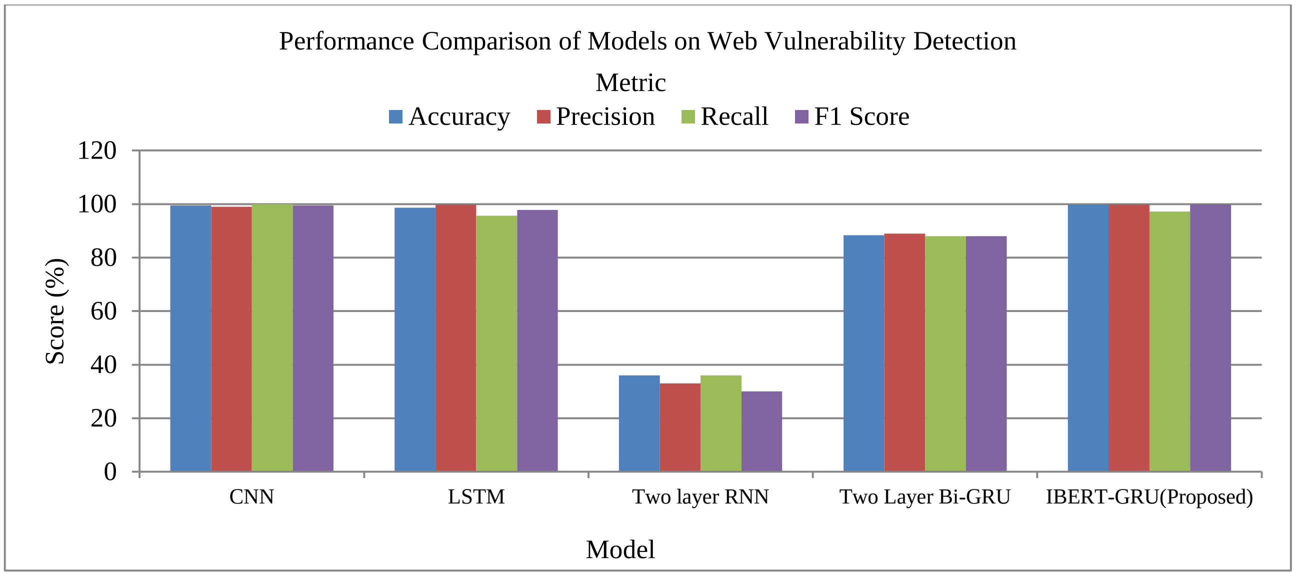


Fig. 7 Performance comparison of the model on web vulnerability detection

4.4.3. Performance Comparison of Attack Detection Times In Various Models

Table 4. Performance comparison of attack detection times in various models

Studies	**Detection Time**
Web attack detection via autoencoder [24]	5.1
Text analysis-based SQLI attack detection [27]	0.89
NLP Technique [10]	0.4
Proposed Model	0.45

Table 4 presents various studies with respect to detection time. In existing studies, web attack detection using an autoencoder takes 5.1 seconds, Text analysis-based SQLI attack detection takes 0.89 seconds, and the NLP technique takes 0.4 seconds. The proposed model should be considered for a detection time of 0.45, given its high-level accuracy, and compared to the previous model's detection time of 5.1, based on accuracy.

4.5. Limitations of the Proposed Model

The proposed model possesses strong contextual learning capabilities; however, it also has several limitations. Deep learning Models suffer from limited interpretability; they face various challenges related to security analysis, which is necessary to justify the model's decisions. With a detection time of 0.45, the proposed model surpasses the previous model in detection time, aiming to achieve a higher level of accuracy.

4.6. Advantages of the Proposed Model

The proposed model is expected to offer several advantages in the domain of web vulnerability detection. It is the powerful and contextual ideas about IBERT that accurately capture the semantic relationship among the patterns with complex types of web requests. It is also enabled to detect attacks of SQL injection, cross-site scripting, and command injection. Adding a GRU-based classifier enhances sequential pattern recognition by preserving long-term dependencies and the temporal relationship between input tokens. This helps look at multi-step attacks. This framework also talked about how automated detection with low values could be a very effective way to keep online environments safe. The proposed model, IBERT-GRU, should be able to grow, learn, and utilize language to address web security issues.

5. Conclusion and Future Work

Web vulnerabilities tend to be a significant threat to digital infrastructures, resulting in data breaches, financial losses, and violations of privacy. Traditional detection lags in addressing the evolving nature of attacks and the complex architectures of modern and evolving web systems. The proposed deep learning-based IBERT-GRU model could be more efficient in detecting online vulnerabilities. IBERT offers deep contextual embedding from online inputs, making it optimal for GRU to capture sequential dependencies, while providing examination of syntax, web traffic, and code structure behavior.

While testing the developed model using datasets from OWASP, Kaggle, and CSIC-2010, the model achieved an accuracy of 99.9%, a Precision of 97.2%, and a recall of 99.85%. The F1 score of the model was 99.85%, which outperformed the traditional deep learning models, CNN and LSTM. The proposed model exhibits a significantly improved computational speed compared to baseline models, with a 6-minute reduction in inference time to 150 milliseconds.

Deep learning Models remain a "black box," critical in high-stakes security situations. Adding proven models like SHAP and LIME helps explain decisions to cybersecurity experts. The performance of the model is enhanced by incorporating multilingual and cross-platform vulnerability databases, making the model more adaptable to the online ecosystem. False negative of the model is reduced by training the model in real-time traffic and with different network conditions. The proposed work could be further enhanced by adding reinforcement learning or GNN for an adaptive mitigation engine that learns new attack techniques. Finally, lightweight deployment with edge computing enables framework analysis in low-resource settings, such as IoT applications.

References

[1] Subhadeep Chell et al., "Real-Time Threat Detection and Mitigation in Web API Development," *2024 International Conference on Electrical Electronics and Computing Technologies (ICEECT)*, Greater Noida, India, pp. 1-9, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[2] Qui Cao et al., "BERT-Enhanced DGA Botnet Detection: A Comparative Analysis of Machine Learning and Deep Learning Models," *2024 13th International Conference on Control, Automation and Information Sciences (ICCAIS)*, Ho Chi Minh City, Vietnam, pp. 1-6, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[3] Shalini Verma, and Harish Kapoor, "Machine Learning for Predictive Maintenance: A Cloud Computing Architecture and Lessons for a Healthcare Context," *International Academic Journal of Science and Engineering*, vol. 8, no. 2, pp. 1-5, 2021. [Publisher Link]

[4] Rania Zaimi et al., "An Enhanced Mechanism for Malicious URL Detection using Deep Learning and DistilBERT-based Feature Extraction," *The Journal of Supercomputing*, vol. 81, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[5] Sofonias Yitagesu et al., "Systematic Literature Review on Software Security Vulnerability Information Extraction," *ACM Transactions on Software Engineering and Methodology*, pp. 1-51, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[6] Ihsan Ullah et al., "Unveiling the Power of Deep Learning: A Comparative Study of LSTM, BERT, and GRU for Disaster Tweet Classification," *IEIE Transactions on Smart Processing & Computing*, vol. 12, no. 6, pp. 526-534, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[7] Abba Suganda Girsang, and Stanley, "Hybrid LSTM and GRU for Cryptocurrency Price Forecasting Based on Social Network Sentiment Analysis Using FinBERT," *IEEE Access*, vol. 11, pp. 120530-120540, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[8] Sarbast H. Ali et al., "Web Vulnerabilities Detection Using a Hybrid Model of CNN, GRU and Attention Mechanism," *Science Journal of University of Zakho*, vol. 13, no. 1, pp. 58-64, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[9] Vahid Babaey, and Hamid Reza Faragardi, "Detecting Zero-Day Web Attacks with an Ensemble of LSTM, GRU, and Stacked Autoencoders," *Computers*, vol. 14, no. 6, pp. 1-29, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[10] Yunus Emre Seyyar, Ali Gökhan Yavuz, and Halil Murat Ünver, "An Attack Detection Framework based on BERT and Deep Learning," *IEEE Access*, vol. 10, pp. 68633-68644, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[11] Abdu Salam et al., "Deep Learning Techniques for Web-Based Attack Detection in Industry 5.0: A Novel Approach," *Technologies*, vol. 11, no. 4, pp. 1-18, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[12] Prabhuta Chaudhary, Ayush Verma, and Manju Khari, *Harnessing Language Models and Machine Learning for Rancorous URL Classification*, 1st ed., Cybersecurity and Data Science Innovations for Sustainable Development of HEICC, pp. 273-288, 2025. [Google Scholar] [Publisher Link]

[13] Sidwendluian Romaric Nana et al., "Deep Learning and Web Applications Vulnerabilities Detection: An Approach based on Large Language Models," *International Journal of Advanced Computer Science & Applications*, vol. 15, no. 7, pp. 1-9, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[14] Refat Othman, Bruno Rossi, and Barbara Russo, "A Comparison of Vulnerability Feature Extraction Methods from Textual Attack Patterns," *2024 50th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Paris, France, pp. 419-422, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[15] Ioana Branescu, Octavian Grigorescu, and Mihai Dascalu, "Automated Mapping of Common Vulnerabilities and Exposures to MITRE ATT&CK Tactics," *Information*, vol. 15, no. 4, pp. 1-19, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[16] Jianing Liu et al., "Enhancing Vulnerability Detection Efficiency: An Exploration of Light-Weight LLMs with Hybrid Code Features," *Journal of Information Security and Applications*, vol. 88, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[17] Jaydeep R. Tadhani et al., "Securing Web Applications against XSS and SQLi Attacks using a Novel Deep Learning Approach," *Scientific Reports*, vol. 14, pp. 1-17, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[18] Remzi Gürfidan, "VULREM: Fine-Tuned BERT-Based Source-Code Potential Vulnerability Scanning System to Mitigate Attacks in Web Applications," *Applied Sciences*, vol. 14, no. 21, pp. 1-14, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[19] Soolin Kim et al., "Vuldebert: A Vulnerability Detection System using Bert," *2022 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, Charlotte, NC, USA, pp. 69-74, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[20] Mudassor Ahmed Chowdhury, Mushfiqur Rahman, and Sifatnur Rahman, "Detecting Vulnerabilities in Website using Multiscale Approaches: Based on Case Study," *International Journal of Electrical & Computer Engineering*, vol. 14, no. 3, pp. 2814-2821, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[21] Sachin Kumar Sharma et al., *Web Security Vulnerabilities: Identification, Exploitation, and Mitigation*, 1st ed., Cybersecurity CRC Press, pp. 183-218, 2021. [Google Scholar] [Publisher Link]

[22] Serena Elisa Ponta, Henrik Plate, and Antonino Sabetta, "Detection, Assessment and Mitigation of Vulnerabilities in Open Source Dependencies," *Empirical Software Engineering*, vol. 25, pp. 3175-3215, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[23] Cho Xuan Do, Nguyen Trong Luu, and Phuong Thi Lan Nguyen, "Optimizing Software Vulnerability Detection using RoBERTa and Machine Learning," *Automated Software Engineering*, vol. 31, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[24] Jiancong Li et al., "Web Application Attack Detection Based on Attention and Gated Convolution Networks," *IEEE Access*, vol. 8, pp. 20717-20724, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[25] Hieu Mac et al., "Detecting Attacks on Web Applications Using Autoencoder," *Proceedings of the 9th International Symposium on Information and Communication Technology*, Danang City Viet Nam, pp. 416-421, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[26] Vijaya Arjunan et al., "Deciphering Ancient Tamil Epigraphy: A Deep Learning Approach for Vatteluttu Script Recognition," *Journal of Internet Services and Information Security*, vol. 15, no. 1, pp. 451-467, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[27] Lu Yu, Senlin Luo, and Limin Pan, "Detecting SQL Injection Attacks based on Text Analysis," *3rd International Conference on Computer Engineering, Information Science & Application Technology*, pp. 95-101, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[28] Geetha Krishna Venkatesh Maroju, and Sai Nandu Posina, "*Comparative Analysis of LSTM, GRU, and BERT Models for Fake News Detection*," Bachelor Thesis, Blekinge Institute of Technology, pp. 1-72, 2025. [Google Scholar] [Publisher Link]

[29] S. Poornimadarshini et al., "Bibliometric Analysis of IJISS Journal based on Citation and Publication Relevant Metrics," *Indian Journal of Information Sources and Services*, vol. 14, no. 4, pp. 153-158, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[30] Ouissem Ben Fredj et al., "An OWASP Top Ten Driven Survey on Web Application Protection Methods," *Risks and Security of Internet and Systems: 15th International Conference, CRiSIS 2020, Paris, France*, pp. 235-252, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[31] Maysoon Khazaal Abbas Maaroof, and Med Salim Bouhlel, "Drone Image Localization by Faster R-CNN Algorithm and Detection Accuracy," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 16, no. 1, pp. 172-189, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[32] Rokia Lamrani Alaoui, and El Habib Nfaoui, "Deep Learning for Vulnerability and Attack Detection on Web Applications: A Systematic Literature Review," *Future Internet*, vol. 14, no. 4, pp. 1-46, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[33] Guanjun Lin et al., "Software Vulnerability Detection using Deep Neural Networks: A Survey," *Proceedings of the IEEE*, vol. 108, no. 10, pp. 1825-1848, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[34] Saif ur Rehman et al., "DIDDOS: An Approach for Detection and Identification of Distributed Denial of Service (DDoS) Cyberattacks using Gated Recurrent Units (GRU)," *Future Generation Computer Systems*, vol. 118, pp. 453-466, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[35] Muhammad Noman, Muhammad Iqbal, and Amir Manzoor, "A Survey on Detection and Prevention of Web Vulnerabilities," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 6, pp. 1-20, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[36] A.M.B. Mohamad et al., "Impact of using Website on Online Learning Behavior," *International Academic Journal of Social Sciences*, vol. 5, no. 2, pp. 76-90. 2018. [CrossRef] [Google Scholar] [Publisher Link]

[37] Nicolás Montes et al., "Web Application Attacks Detection using Deep Learning," *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pp. 227-236, 2022. [CrossRef] [Google Scholar] [Publisher Link]